VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vyhledávání v kolekci obrázků pomocí komprese
Searching in Images Collection Based on Compression

2012 **Bc. Jakub Říman**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Říman**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Vyhledávání v kolekci obrazků pomocí komprese
Searching in Images Collection Based on Compression

Zásady pro vypracování:

Hledání podobnosti mezi obrázky je jedním z aktuálních problémů v oblasti Information Retrieval. V rámci této diplomové práce prostuduje diplomant přístup k hledání podobnosti mezi obrázky pomocí kompresních algoritmů. Nedílnou součástí k porovnávání nějakých informací jsou samotná data. V rámci práce se bude jednat o data získaná pomocí API jednoho z významných online zdrojů.

Jednotlivé části práce jsou:
1. Prostudovat přístup k vyhledání obrázku popsaných Geraldem Schaeferem.
2. Implementovat vybrané postupy pro vyhledání obrázků na základě komprese.
3. Implementovat stahování obrázků z vybraného online zdroje.
4. Provedení experimentů a vizualizace získaného výsledku.
5. Zhodnocení experimentů na přesnost a rychlost.
6. Popsání možných dalších rozšíření vyvinuté metody.

Seznam doporučené odborné literatury:

Schaefer, G., 2010. Content-based retrieval from image databases: Colour, compression, and browsing. 2010 International Conference on Information Retrieval Knowledge Management CAMP, p.5-10. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5466891.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne 30. 4. 2012 v Ostravě

.....

Bc. Jakub Říman

Poděkování

Rád bych poděkoval Ing. Janu Martinovičovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Vyhledávání a porovnávání obrázků je jedno z aktuálních témat v oblasti získávání informací. Z důvodů stále vzrůstajícího množství obrázků na internetu i v lokálních úložištích se upouští od přístupu vyhledávání v databázích obrázků na základě textové anotace a využívají se efektivnější metody vyhledávání obrázků na základě obsahu. Diplomová práce se zabývá právě metodami vyhledávání na základě obsahu obrázku. Jelikož je většina obrázků již v komprimované podobě, je práce zaměřena na vyhledávání informací z komprimovaných dat pomocí vektorové kvantizace. Cílem diplomové práce je získání obrázků z online zdrojů pro experimenty a implementace systému pro indexaci včetně vyhledávání obrázků. V závěru práce jsou popsány dosažené výsledky a prezentovány návrhy pro zefektivnění systému.

Klíčová slova

Získávání dat, vyhledávání na základě obsahu obrázku, Flickr, vektorová kvantizace

Abstract

Searching and comparing images is one of actual topic in information retrieval domain. Because of a huge amount of images on the internet and local storages, annotation based approach is decreasing. In my thesis, I have been working on more effective methods such as content based image retrieval. Because of the most of the images are in compressed format I am working with information retrieval on compressed data by vector quantization. The main goal of this thesis are retrieving images to benchmark and implementing system for indexation and searching images. At the end I am describing achieved results and I suggesting some ideas to improve the system.

Keywords

Information retrieval, content based image retrieval, Flickr, vector quantization

Seznam použitých zkratek a symbolů

CBIR	- Content-based image retrieval, vyhledávání na základě obsahu obrázků
VQ	- Vector quantization, vektorová kvantizace
API	- Application programming interface, rozhraní pro programování aplikací
IR	- Information Retrieval, získávání informací

Obsah

Seznam obrázků

Seznam tabulek

„A picture is worth a thousand words“[1]

[Obrázek řekne víc než tisíc slov]

[1] Fred R. Barnard, Printers' Ink, March 10, 1927

1. Úvod

Ve 21. století, v době digitalizace, kdy se snažíme o maximální dostupnost informací, jsou datová centra plná textových a multimediálních dat. Současně roste zájem o efektivní vyhledávání konkrétních informací z těchto zdrojů, proto se upouští od staršího a méně efektivního přístupu vyhledávání na základě textové anotace. Pokud se budeme věnovat pouze obrázkům, zjistíme, že existují metody pro tzv. vyhledávání na základě obsahu obrázku, což znamená, že získávání informací z obrázků je založeno na analýze obrázku bez použití *metadat* jako klíčových slov, či textových anotací. Tyto metody obvykle čerpají obsah, tedy vlastnosti obrázků, jako například barvu nebo tvar z nekomprimovaných dat. Avšak většina obrázků dostupných na internetu je uložena v komprimované formě, kvůli úsporám v kapacitách úložiště a snížení velikosti přenosu dat. Dekomprimace dat je navíc časově náročná operace.

Má diplomová práce prezentuje jiný přístup – získávání informací přímo z komprimovaných dat bez nutnosti jejich dekomprimace. Tuto techniku popsal Gerald Schaefer v publikaci Content-based Retrieval of Compressed Images [11]. G.Schaefer v textu prezentuje dvě řešení – získávání dat na stávající kompresní technice tzv. vektorové kvantizaci a vývoj vlastního tzv. 4-kritériového kompresního algoritmu, kde jsou data v komprimované podobě stále přímo vizuálně smysluplná a mohou být využita pro získávání informací.

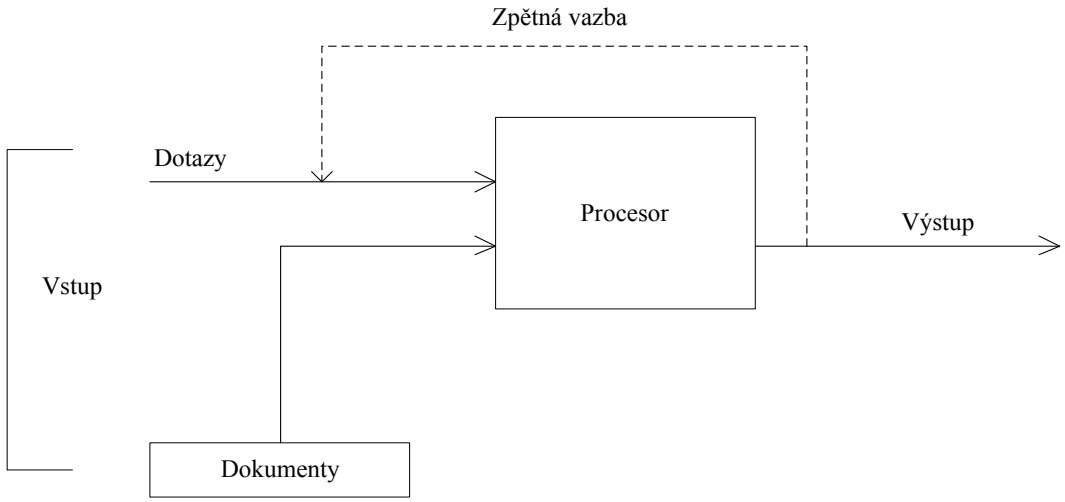
1.1. Struktura práce

Práce je rozdělena do šesti kapitol. V kapitole 2 je prezentován úvod do teorie vyhledávání informací, jsou zde vysvětleny rozdíly mezi vyhledávacími metodami a popsány oblasti využití, včetně ukázky stávajících řešení využívajících vyhledávání obrázků. Kapitola 3 je zaměřená na vysvětlení vybraných metod a algoritmů (K-means, LBG, VQ) použitých v implementaci. V kapitole 4 je uveden popis samotné implementace. Kapitola 5 obsahuje experimenty a jejich výsledky. Shrnutí výsledků práce je prezentováno v závěru (kapitola 6).

2. Úvod do teorie vyhledávání informací

Oblast vyhledávání informací (ang. Information Retrieval) je zaměřena na vyhledávání dokumentů a vyhledávání v obsahu dokumentů. Vyhledávání informací je založeno na studiu několika oborů, informačních technologií, matematiky, knihovnictví, statistiky a dalších. Mnoho univerzit a knihoven využívá systémy pro vyhledávání informací k poskytování služeb, avšak nejznámější aplikace vyhledávání informací jsou internetové vyhledávače [6].

Proces vyhledávání informací začíná uživatelským dotazem do systému. Dotaz je určité vyjádření toho, co potřebujeme od systému vědět, konkrétní příklad je zadání textového řetězce ve webovém vyhledávači. Při vyhledávání informací dotaz není unikátně identifikován právě jedním objektem v systému, ale výsledkem může být zároveň několik objektů s různou hodnotou relevance. Do procesu vyhledávání vstupují dokumenty (*obrázek č. 1*) jako vstupní data, ze kterých se získávají výsledky pro daný dotaz.



Obrázek 1: Schéma typického systému pro vyhledávání informací[2]

[2] http://www.dcs.gla.ac.uk/Keith/Chapter.1/Ch.1.html

2.1. Obrazové vyhledávání

Termín obrazové vyhledávání (ang. Image Retrieval) se začal používat v sedmdesátých letech 20. století s rostoucím objemem digitálních obrázků [2]. Metody pro obrazové vyhledávání můžeme rozdělit na dvě fáze - získávání vlastností z obrázku a měření podobnosti mezi získanými vlastnostmi obrázku [10]. Také existují dva základní přístupy k obrazovému vyhledávání - vyhledání na základě textové anotace a vyhledávání na základě obsahu obrázku.

2.2. Vyhledávání na základě textové anotace

Při vyhledávání na základě textové anotace (ang. Text-Based Image Retrieval nebo také Annotation-Based Image Retrieval) se k obrázku vloží textový popis. Vyhledávání probíhá pouze na základě informací z textových dat, obsah obrázku se ignoruje [6]. Hlavní nevýhoda tohoto přístupu je subjektivita lidského pohledu na obrázek při vytváření textového popisu. Ruční popisování obrázků je pro velké databáze velmi nákladné a nemusí zachycovat všechna klíčová slova pro popis obrázku. Také chybí jakákoliv automatizace procesu. Aktuální webové prohlížeče vyhledávají výhradně jen s využitím textové anotace, což mnohdy produkuje nerelevantní výsledky, například u vícevýznamových slov.

2.3. Vyhledávání na základě obsahu obrázků

Vyhledávání na základě obsahu obrázků (ang. Content-Based Image Retrieval, totožný s pojmem Content-Based Visual Information Retrieval, dále jen CBIR) se začalo rozvíjet začátkem 90. let [11]. CBIR umožňuje vyhledávat obrázky bez nutnosti používat klíčová slova či textové anotace. Tato metoda je založena na získávání a porovnávání charakteristik obrázků, jako jsou například barva, textura či tvar. Nejběžněji metody pro porovnání dvou obrázků (vzor a obrázek z databáze) v CBIR využívají některé z metrik pro výpočet vzdálenosti obrázků. Tato vzdálenost obrázků reprezentuje podobnost dvou obrázků na základě již zmíněných charakteristik.

Základní charakteristiky, které můžeme vyčíst z obrázku, jsou:

Barva

Zkoumání obrázků na základě barev, které obsahují, je jedna z nejrozšířeněji používaných technik, protože nezáleží na velikosti obrázku či jeho orientaci [6]. Porovnávání dvou obrázků na barevnou podobnost je dosaženo počítáním barevných histogramů. Pro každý obrázek poté vypočteme vzdálenosti těchto histogramů.

Textura

Textura popisuje prostorově opakující se struktury povrchu, utvořené opakující se specifickým prvkem nebo několika prvky, v různém prostorovém umístění [3]. Obecně opakování zahrnuje obměny měřítka, orientace, či dalších geometrických a optických vlastností prvku. Mnoho statistických rysů textury je založených na kookurenční matici, která ukazuje, jak často se vyskytuje každý jednotlivý pár obrazových bodů (pixelů) v úrovních šedé. Do skupiny příznaků počítaných z textury patří např. kookurenční matice nebo korelogramy.

Tvar

Tvarem nemyslíme přímo tvar obrázku, ale nějakou významnou oblast se specifickou vlastností. Pro získání oblastí využíváme algoritmů pro detekci hran v obrázku. V některých případech rozpoznání tvarů potřebujeme lidský zásah, protože metody pro segmentaci tvaru jsou příliš složité k úplné automatizaci. Algoritmy pro rozpoznávání tvarů by měly být invariantní pro rotaci, posun a zvětšení tvaru v obrázku [6].

Barevný histogram

Barevný histogram je reprezentací rozložení barev v obrazu. Histogram je měřítkem statistického popisu frekvenčního rozložení ve vztahu k výskytu frekvencí různých tříd barev. Třídy barev jsou určité oblasti v barevném prostoru. Každá třída barev v obrazu má svůj index, kterým se při výpočtu histogramu násobí. Pokud je v obraze zastoupena nejvíce modrá, neznamená to, že výsledný histogram bude odpovídat hlavně hodnotě modré barvy. Je to způsobeno tím, že lidské oko má pro různé barvy různou citlivost. Hodnoty pro jednotlivé barevné složky jsou proto obvykle stanoveny následujícím způsobem: červená - 0,3; zelená - 0,59; modrá - 0,11.

Hlavní nevýhodou histogramu v použití v systémech pro vyhledávání obrázků na základě obsahu je, že reprezentace je závislá na barvě objektu, s kterým se pracuje a že ignoruje jeho tvar, texturu a prostorové uspořádání. Jinými slovy, nepoznáme, jestli se jedná o žlutou slunečnici nebo žlutý talíř [5].

Výzkum v CBIR se snaží nalézt nejlepší řešení pro tyto problémy:

- správné porozumění uživatelskému dotazu;
- identifikace všech možných prostředků k popisu obrázku;
- extrakce obrazových vlastností z obrázku;
- porovnávání obrázků reflektující lidský názor a vnímání obrázku;
- prezentace výsledků pro uživatele.

2.4. Oblasti využití a existující řešení využívající CBIR

Vyhledávání a porovnávání obrázků má široké využití skrz mnoho oborů, v lékařství to jsou například analýzy snímků z magnetické rezonance či snímků z ultrazvuku. Data jsou uchovávána a mohou sloužit k pozdějšímu výzkumu a vzdělávacím účelům. Mezi další využití CBIR patří filtrování a cenzura obrázků s kontroverzními výsledky, rozpoznávání prvků ochrany duševního vlastnictví nebo předpověď počasí na základě porovnávání aktuálních dat s minulostí. Nejlépe vyvinuté metody CBIR jsou aplikovány ve vojenství, například rozpoznávání nepřátelských letadel na radarech, identifikace cílů ze satelitních snímků, policie a armáda také využívají CBIR pro identifikaci osob podle obličeje, otisků prstů či DNA a mnohé další techniky pro prevenci kriminality. Londýnská metropolitní policie je zapojena do projektu vytváření databáze fotek odcizených předmětů[3]. Vysílací společnosti a reklamní agentury by se neobešly bez správy fotografií či vyhledávání v archivech obsahujících miliony hodin videa. Používají vlastní databáze obrázku pro ilustrace knih nebo například článků v novinách [6]. V domácnostech bychom mohli v budoucnu využít správu fotografií např. pro dotazy typu „vyhledej fotky z pláže v Chorvatsku“.

[3] http://www.arttic.com/grasp/

Výběr z aktuálních systémů pro vyhledávání obrázků dostupných v roce 2012.

TinEye

TinEye[4] je bezplatný online vyhledávač obrázků, který na základě vložení URL adresy obrázku či po nahrání obrázku z vlastního počítače dokáže najít další výskyty obrázku na webu. V září 2011 server TinEye ohlásil[5], že indexuje více než 2 miliardy obrázků pro porovnávání. Avšak tento počet je stále zanedbatelný k celkovému množství obrázků dostupných na internetu.

QBIC

QBIC (Query By Image Content) patří mezi nejznámější systémy s obrazovým vyhledáváním, vyvinutý společností IBM. QBIC podporuje jak dotazy na základě náčrtu z palety barev, tak dotazy na zadaný obrázek. QBIC pro vyhledávání používá barevné histogramy v několika barevných prostorech, momentové invariant, ale také textový popis klíčovými slovy. Pro indexaci obrázků používá R*-strom[6], metrikou pro vyhledávání je vážená euklidovská vzdálenost. Vývoj QBIC je již ukončen, využívá jej stále například petrohradské muzeum Ermitáž[7].

Tiltomo

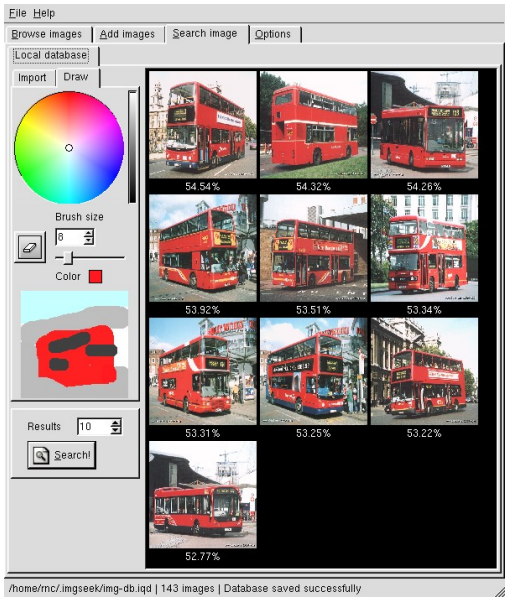
Tiltomo[8] je systém umožňující vizuální vyhledávání v databázi obsahující obrázky ze serveru Flickr. Systém Tiltomo si neukládá obrázky na vlastních serverech, pouze odkazuje na umístění na serveru Flickr. Uživatel si může zvolit, zda chce vyhledávat obrázky s podobným tématem (objekt + barva + textura) či obrázky s podobnými příznaky (100% barva + textura).

imgSeek

ImgSeek[9] je kolekce open source projektů pro vyhledávání obrázků na základě vizuální podobnosti. Dotazy mohou být ve formě hrubě nakresleného náčrtku nebo přímo jako obrázek

[4] http://www.tineye.com/
[5] http://www.tineye.com/updates?year=2011/
[6] http://www.cis.temple.edu/~vasilis/Courses/CIS750/Papers/qbic.pdf
[7] http://www.hermitagemuseum.org/fcgi-bin/db2www/qbicSearch.mac/qbic?selLang=English
[8] http://www.tiltomo.com/
[9] http://www.imgseek.net/

nahraný z počítače. ImgSeek vyhledává podle barvy, tvaru, metadat a dalších příznaků. Aplikace zobrazí 10 nejlepších výsledků z kolekce tisíců obrázků.

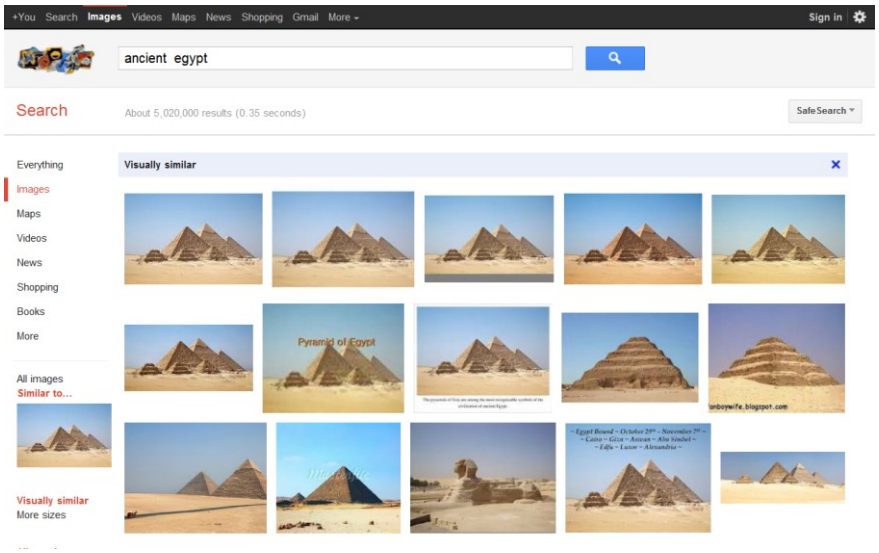


Obrázek 2: Aplikace s vyhledáním výsledků podle náčrtu

Google Image Search

Google Image Search[10] je služba vyhledávající obrázky, spuštěna v roce 2001. Obrázky jsou vyhledávány podle okolního textu, tzn., nejedná se o vyhledávání na základě obsahu. Nicméně bylo představeno rozšíření aplikace o funkci Google Similar Images, kdy k vyhledávanému obrázku přibyl odkaz „Similar images“. Takto zobrazené výsledky už používají algoritmy pro vyhledávání v obsahu obrázku.

[10]http://images.google.com/



Obrázek 3: Google similar images

2.5. Flickr.com

Součástí diplomové práce jsou experimenty, pro které bylo nutné vytvoření lokální databáze obrázků získaných z online zdrojů. Pro implementaci byl vybrán populární server Flickr. Webová služba Flickr[11] je komunitní web pro sdílení fotografií na internetu. Flickr byl spuštěn v roce 2004 a byl jedním z prvních serverů Web 2.0, který umožňoval používat štítky (tagy) s popisem či geolokací. Flickr umožňuje zvolit, kdo bude mít oprávnění zobrazit fotografii (veřejné, soukromé, jen přátelé). Také lze nastavit, pod jakou licencí bude snímek sdílen. Mnoho snímků je šířeno pod licencí umožňující další použití, Flickr tedy můžeme využívat jako fotobanku snímků. V srpnu 2011 bylo oznámeno[12], že Flickr sdílí více než 6 miliard obrázků a registrováno je 51miliónů uživatelů. Pro uživatele mobilních zařízení existují oficiální aplikace pro operační systémy iOS, Windows Phone 7 a Android.

2.6. Barevné modely

Pro práci s obrázky je potřeba pochopit, jak digitální fotoaparáty obrázky ukládají. Barevný model popisuje základní barvy a určuje způsob mísení těchto základních barev do

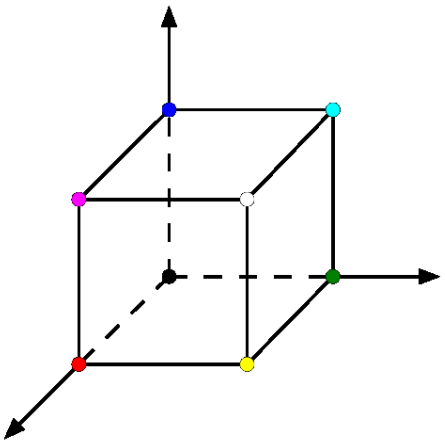
[11] http://www.flickr.com/
[12] http://news.softpedia.com/news/Flickr-Boasts-6-Billion-Photo-Uploads-215380.shtml

barvy výsledné. V přírodě je barva dána směsí světla různých vlnových délek a různé druhy barevných modelů se snaží napodobit barvu co nejvěrněji [5].

- Aditivní míchání barev - složky barev se sčítají a výsledek je světlo větší intenzity. Aditivní skládání barev pracuje se třemi základními barvami: červenou, zelenou a modrou. Využívají jej např. digitální fotoaparáty, monitory nebo projektory.
- Subtraktivní míchání barev - s každou přidanou barvou se ubírá část původního světla - světlo prochází jednotlivými barevnými vrstvami a je stále více pohlcováno. Základní barvy jsou azurová, purpurová a žlutá; smícháním všech těchto barev vznikne černá. Tohoto barevného modelu využívají například tiskárny.

2.6.1. RGB barevný model

Model RGB (Red, Green, Blue – červená, zelená, modrá) vychází z faktu, že lidské oko obsahuje tři základní druhy buněk citlivých na barvu. Tyto buňky jsou citlivé na elektromagnetické záření vlnové délky, které zhruba odpovídají červenému světlu (700 nm), zelenému (546,1 nm) a modrému (435,8 nm) [5]. Kombinací těchto tří barev lze získat téměř všechny barvy barevného spektra, pokud použijeme pro každou barvu 8bitů, lze vyjádřit 16,7 milionů barev.



Obrázek 4: Reprezentace barevného prostoru RGB

2.7. Komprimace dat

Komprimace dat (také komprese dat) je postup ukládání dat, kdy hlavní snahou komprimace je zmenšit velikost datového souboru nebo zmenšit potřebu zdrojů při ukládání informací, např. snížení doby nutné pro přenos. Hlavní nevýhoda je výpočetní náročnost některých kompresních algoritmů. Při volbě kompresních algoritmů sledujeme celou řadu důležitých parametrů, avšak klíčový parametr je ztrátovost použité metody. Podle ztrátovosti dělíme kompresní metody na dva typy:

- bezeztrátové kompresní metody;
- ztrátové kompresní metody.

Bezeztrátové kompresní algoritmy umožňují získat přesná originální data z rekonstruovaných komprimovaných dat, na rozdíl od ztrátové komprese, kde se rekonstruovaná data pouze přiblíží k původnímu originálu.

Princip ztrátových kompresních algoritmů u obrázků využívá skutečnosti, že obraz obsahuje informace, které i přes jejich ztrátu nebudou mít vliv na vnímání obrázku. Míra komprese je v algoritmech proměnná a lze podle potřeb nastavit. Algoritmus vektorové kvantizace, který používám ve svém vypracování je příklad ztrátového kompresního algoritmu. Hlavní výhody komprese jsou snížení času a ceny přenosu dat [20].

Další formáty pro rastrovou grafiku využité v diplomové práci jsou JPEG, TIFF a PNG.

2.7.1. JPEG

Komprese JPEG (Joint Photographic Expert Group) je nejčastěji používaný formát pro ukládání fotografií, řadí se mezi ztrátové kompresní algoritmy. Každým uložením dochází ke ztrátě dat. JPEG umožňuje nastavení míry komprese, komprese je založená na separaci jasové a barvonosné složky obrazu, na které je aplikovaná diskrétní kosinová transformace.

2.7.2. TIFF

TIFF (Tagged Image File Format) je formát pro ukládání rastrové grafiky, umožňující bezztrátovou kompresi s využitím Huffmanova kódování. Obrázek ve formátu TIFF tedy lze

editovat a znovu ukládat bez ztráty obrazových informací. TIFF umožňuje jako jeden z mála grafických formátů vícestránkové soubory.

2.7.3. PNG

Grafický formát PNG (Portable Network Graphics) umožňuje bezztrátovou kompresi, byl navrhnut jako náhrada za formát GIF (Graphics Interchange Format). PNG nabízí podporu 24 bitové barevné hloubky. Formát PNG nepodporuje barevný prostor CMYK.

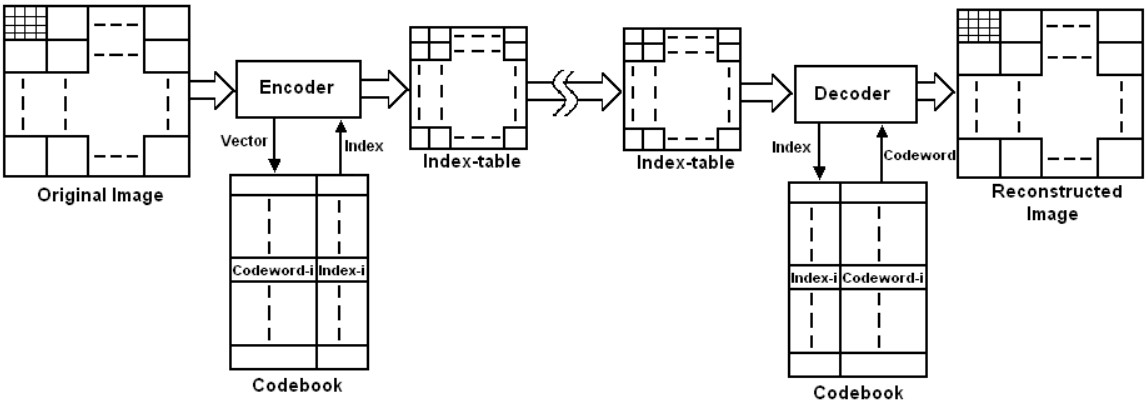
3. Vybrané metody a algoritmy

Vyhledávání na základě obsahu obrázků pomocí vektorové kvantizace využívá faktu, že kódová kniha obsahuje přesný popis obrázku, obrázky tedy mohou být porovnávány v komprimované formě podle informací uložených přímo v jejich kódových knihách [11]. Ačkoliv vektorová kvantizace patří mezi ztrátové kompresní algoritmy, data po zkomprimování obsahují nejen barevné informace, ale také prostorové informace (ang. Spatial Information) jako jsou textura a tvar. Prostorové informace jsou dostupné díky tomu, že obrázek je rozdělen do bloků a bloky jsou kódovány jako celek (*obrázek 17*). Vytváření nové trénovací knihy pro každý obrázek zajistí, že se vytvoří kódové knihy nejlépe přizpůsobené vstupnímu obrázku.

3.1. Vektorová kvantizace

Vektorová kvantizace (ang. Vector Quantization, dále VQ) je metoda využívaná v aplikacích s kompresí obrázků, zvuků či videa. Tato ztrátová kompresní technika rozděluje vektorový prostor do několika oblastí. Pro každou oblast je vytvořen vektor, který nejlépe reprezentuje danou oblast, tzv. *centroid*. V mé diplomové práci pracuji s obrázky rozdělenými do oblastí 4 x 4 obrazové body (*pixely*). Vektor obsahuje 48prvků (4 x 4 x 3), jelikož nese informaci o šestnácti obrazových bodech z obrázku, každý obrazový bod obsahuje tři hodnoty (vyjádření barvy v RGB prostoru). Vektory tvoří nepravidelné shluky oblastí, z nichž se metodami (nejčastěji používaná metoda se nazývá K-means) založenými na průměrování stanovují *centroidy*. *Centroid* musí mít ke všem vektorům v oblasti minimální vzdálenost, pro výpočet této vzdálenosti se používá kvadratická míra. *Centroidy* se také označují jako kódová slova, seznam kódových slov tvoří kódovou knihu [11].

Nalezení kódové knihy nejlépe reprezentující danou množinu vektorů je NP-úplný problém. Po vytvoření kódové knihy může být obrázek na vstupu reprezentován pouze indexy příslušných vektorů v kódové knize. Existuje více variant provedení vektorové kvantizace tzv. *Split VQ, Multi-stage VQ, Tree-structured VQ* [14].



Obrázek 5: Diagram procesu vektorové kvantizace [17]

3.2. Metoda nejbližších středů K-means

Algoritmus K-means iterativně hledá hodnoty vektorů tak, že minimalizuje střední odchylku mezi zadanou množinou dat a vektory, které mají k těmto datům nejmenší euklidovskou vzdálenost [13].

Každý shluk je reprezentován středem – *centroidem (obrázek 6)*. Kritériem pro výběr *centroidu* je minimální vzdálenost od středu shluků. Inicializaci kódové knihy můžeme realizovat dvěma způsoby - pomocí zcela náhodných hodnot nebo můžeme vybrat několik vektorů z trénovacích dat a prohlásit je za původní kódovou knihu. Problém u algoritmu K-means je, že ani jedna metoda nezaručuje, že se algoritmus během iterací nezhroutí. Pokud k některému z kódových vektorů není v kódování přiřazen ani jeden vektor, objeví se v rovnici dělení nulou. V implementaci vektorové kvantizace se používá sekvenční databáze, což algoritmus zpomaluje. Lepší řešení by byla implementace s využitím R-stromu.

„K-means algoritmus pro děti“ [8]:

Byla země a N domečků

1. Jednoho dne K králů přijelo do země.
2. Každý domeček byl zabrán nejbližším králem.
3. Společnost po králích chtěla, aby jejich král byl uprostřed vesnice, proto byl trůn krále přesunut do centra vesnic.

4. Král si všiml, že některé domečky jsou nyní k němu blíž a tak domečky zabral, ale některé také ztratil. Takto se opakovaly kroky 2,3,4.
5. Až jednoho dne se králové už dál nemohli přesunovat a tak se usadili a žijí ve svých vesnicích dodnes.

V matematickém zápise, který je popsán v studijních podkladech [4] k algoritmu potřebujeme:

- Inicializovanou kódovou knihu **Y** (0).
- Trénovací vektory $x(1) \dots x(N)$.

Algoritmus provádíme v k iteracích. Inicializaci označíme $k = 0$. Dále budeme potřebovat kritérium pro zastavení iterací ε pro relativní změnu celkového zkreslení D_{VQ}.

- Inicializace: $k = 0$, definujeme **Y**(0).
- Krok 1: nejprve přiřadíme trénovací vektory buňkám – „zakódujeme je:“

  $$Q[x] = y_i(k) \text{ pokud } d\big(x, y_i(k)\big) \leq d(x, y_j(k)) \text{ pro } j \neq i, j \in 1..L$$

  Vektor x pak náleží buňce $C_i(k)$. Celkové zkreslení vypočteme podle rovnice:

  $$D_{VQ} = \frac{1}{N}\sum_{n=1}^{N} d(x(n), Q[x(n)])$$

- Krok 2: Je-li relativní pokles zkreslení

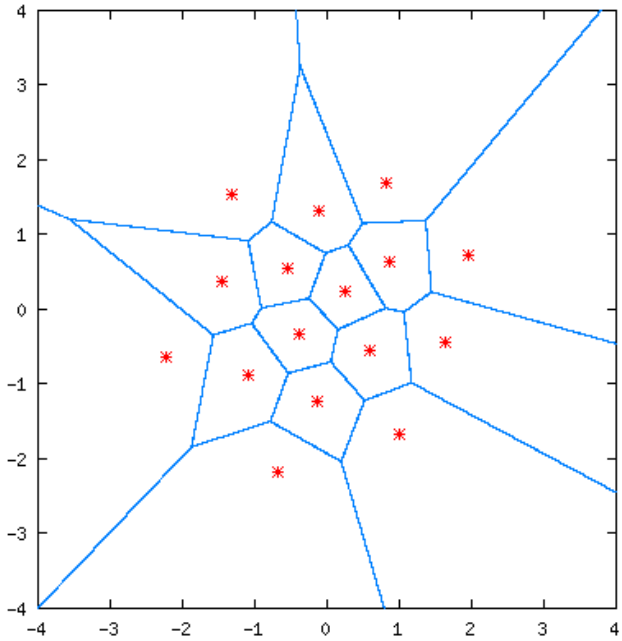
  $$\frac{D_{VQ}(k-1) - D_{VQ}(k)}{D_{VQ}(k)} \leq \varepsilon,$$

  ukončíme algoritmus a prohlásíme k-tou kódovou knihu za výsledek: $\mathbf{Y} = \boldsymbol{Y}(k)$.

- Krok 3: Počítáme nové centroidy každé buňky, uděláme z nich nové kódové vektory:

  $$y_i(k+1) = Cent\big(C_i(k)\big) = \frac{1}{M_i(k)}\sum_{x \in C_i(k)} x,$$

  kde $M_i(k)$ je počet trénovacích vektorů přiřazených buňce i při kódové knize generace k. Dále inkrementujeme: $k = k + 1$, návrat na krok 1.



Obrázek 6: Vektorová kvantizace s vyznačenými centroidy [15]

3.3. LBG algoritmus

Yoseph Linde, Andrés Buzo a Robert M. Gray v roce 1980 navrhli iterativní algoritmus, který řeší problém, jak inicializovat kódovou knihu v algoritmu K-means. Algoritmus LBG postupně dělí shluky v kódové knize [14]. Po každém dělení proběhne další iterace K-means. Velikost kódové knihy se postupně zvětšuje v mocninách dvou. LBG algoritmus je nejčastěji používaný algoritmus pro návrh kódové knihy pro vektorovou kvantizaci. Efektivita LBG algoritmu je extrémně závislá na výběru trénovacích vektorů. Pokud jsou tyto vektory vybrány náhodně kódová kniha má velmi nízkou kvalitu. Pro optimální výsledky je třeba za trénovací vektory použít přímo data určená ke kompresi, tzn. pro každý obrázek budeme vytvářet vlastní kódovou knihu. Tento způsob zvýší kvalitu kódové knihy, ovšem je těžké tento náročný algoritmus provádět v reálném čase.

Problematiku popisuje publikace *Vektorové kvantování* [4], iterace LBG budeme značit $r = 0..R-1$, kde $L = 2^R$.

- Inicializace: $r = 0$, kódová kniha $\mathbf{Y}(0)$ má 1 kódový vektor, který je *centroidem* všech trénovacích vektorů.

- Fáze 1: z kódové knihy o 2^r vektorech uděláme dvakrát větší kódovou knihu (o 2^{r+1} vektorech) tak, že kódové vektory rozštěpíme:

$$y_i(r) \rightarrow \begin{cases} y_{2i-1}(r+1) = y_i(r) + \Delta \\ y_{2i}(r+1) = y_i(r) - \Delta \end{cases},$$

  kde Δ je vektor, kterým od sebe dva nové kódové vektory „odtáhneme“.
- Fáze 2: spustíme *K*-means pro **Y** $(r+1)$.
- Fáze 3: je-li $r+1 = R$, konec, výsledná kódová kniha je **Y** = **Y** $(r+1)$. Jinak zpět na fázi 1.

3.4. Metriky pro porovnání množin

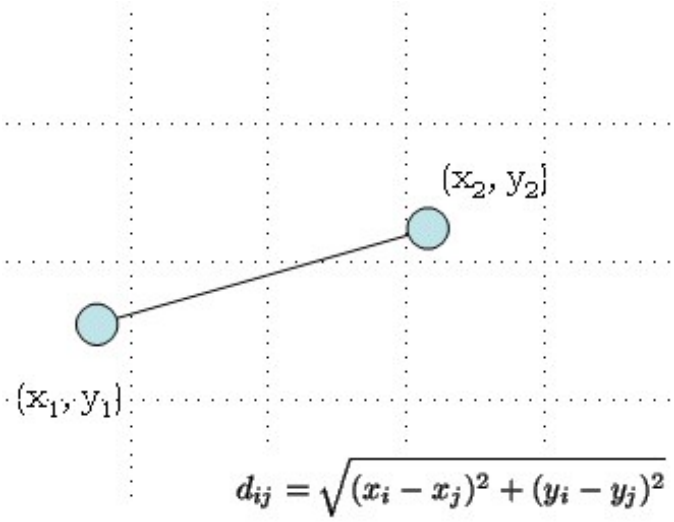
Charakteristiky obrázků jsou uloženy do množin. Množiny je nutné porovnat a zjistit vzájemnou vzdálenost těchto množin. Metrický prostor je matematická struktura, pomocí které lze formálním způsobem definovat pojem vzdálenosti. Na euklidovském prostoru (tj. v rovině, v prostoru, případně ve vícerozměrném prostoru) lze definovat metriku mnoha způsoby, je to například Hausdorffova metrika, Manhattanská metrika, či nejčastěji používaná euklidovská metrika. Měření vzdáleností ve vyhledávání obrázků na základě obsahu znamená, vypočtení vzdálenosti mezi obrázkem položeným jako dotaz a jednotlivými obrázky z databáze. Poté můžeme obrázky seřadit podle vzdálenosti od nejmenší vzdálenosti (největší shoda) po největší vzdálenost (nejmenší shoda).

3.4.1. Eukleidovská metrika

V matematice je euklidovská vzdálenost vyjádřením vzdálenosti mezi dvěma body. Zavedeme-li v n-rozměrném eukleidovském prostoru kartézskou soustavu souřadnic, pak vzdálenost *d* mezi dvěma body *X* a *Y* o souřadnicích $(x_1,x_2,\ldots,x_n)$, $(y_1,y_2,\ldots,y_n)$ je určena vztahem

$$d(x,y) = d(y,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Výsledkem euklidovské metriky je vždy nezáporné reálné číslo. Euklidovskou metriku můžeme vnímat jako formu Pythagorovy věty [18].



Obrázek 7: Euklidovská vzdálenost mezi dvěma body[13]

3.4.2. Hausdorffova metrika

Hausdorffova metrika měří „nejvzdálenějšího nejbližšího souseda“. Pro všechny prvky množiny *A* se spočítají vzdálenosti k nejbližšímu sousedu v množině *B* a výsledkem je maximum těchto hodnot. V matematice měří Hausdorffova metrika neboli Hausdorffova vzdálenost, jak vzájemně vzdálené od sebe jsou dvě podmnožiny metrického prostoru [7]. Pro získání vzdálenosti mezi dvěma body využijeme euklidovskou vzdálenost. Hausdorffova vzdálenost není komutativní, proto $h = (A, B)$ není ekvivalentní s $h = (B, A)$.

Hausdorffova vzdálenost množin *A* a *B* je definována

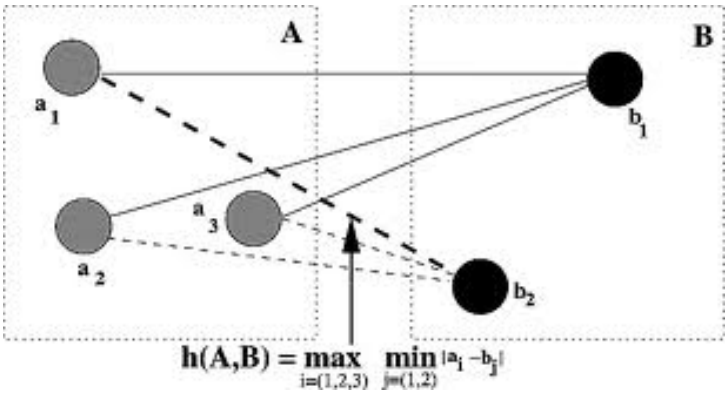
$$d_H(A,B) = \max\big(h(A,B), h(B,A)\big)$$

$$h(A,B) = \max_i \min_j \delta(A_i, B_j)$$

Výhodnější je použití modifikované Hausdorffovy vzdálenosti pro menší vliv odlehlých měření, místo maxima se využívá průměr hodnot.

$$h_{mod}(C_A, C_B) = \frac{1}{N}\sum_{i=1}^{N} min_j \|C_A(i) - C_B(j)\|$$

[13] http://physics.yeditepe.edu.tr/merenturk/tsp_demo/flow_chart.html



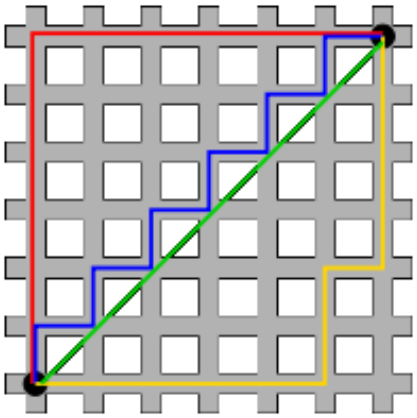
Obrázek 8: Hausdorffova vzdálenost[14]

3.4.3. Manhattanská metrika

Manhattanská metrika (podle pravoúhlého systému ulic na Manhattanu v New Yorku, také nazývána L1 - vzdálenost) je vzdálenost mezi dvěma body měřená podle osy v pravém úhlu. Manhattanská metrika je definována jakou součet absolutní hodnoty rozdílu vrcholů dvou bodů:

$$\rho(x, y) = |x_1 - y_1| + |x_2 - y_2| + \ \ldots + \ |x_n - y_n|$$

Na *obrázku č. 9* zobrazuje zelená linka euklidovskou vzdálenost, červená linka je manhattanská vzdálenost, modrá a žlutá linka mají stejnou vzdálenost jako červená, jen jsou jinak prostorově vyjádřeny.



Obrázek 9: Červená linka zobrazuje manhattanskou vzdálenost[15]

[14] http://www.agu.org/pubs/crossref/2005/2004JD005395.shtml

3.5. Locality sensitive hashing

Kapitola 4.5 popisuje implementaci vyhledávání obrázků s využitím tzv. *Locality sensitive hashing*. Hašování (ang. Hashing) je základní technika v datových strukturách, která využitím matematické funkce (resp. algoritmu), převádí vstupní data do řetězce s pevnou délkou. Výstup hašování se nazývá otisk. Často se používá v kryptografii, například ke kódování hesel [19].

Locality sensitive hashing proti normálnímu hašování umisťuje blízké klíče do blízkých pozic tabulky, což je důležité například při vyhledávaní nejbližšího souseda. V mé implementaci předpokládám, že podobné vektory budou zahašovány pod stejným otiskem.

3.6. Indexační struktury

Pro urychlení a zefektivnění vyhledávání je výhodné použít datovou strukturu, která je přímo určená pro ukládání a práci s vícerozměrnými daty a umožňuje klást prostorové dotazy [1]. R-stromy jsou modifikací B-stromů a jsou používány pro indexování vícerozměrných struktur. Existují varianty těchto stromů například R*-stromy. R-stromy nezaručují dobrý výkon pro nejnepříznivější případ uložení dat, ale jejich implementace dosahují slušných výkonů při použití „reálných“ dat.

V mé diplomové práci je použit sekvenční přístup pro ukládání dat. V doporučené literatuře k diplomové práci [10], ze které jsem vycházel, nebylo uvedeno, jak jsou data uložena. Proto jsem se s využitím R-stromů seznámil až při studiu implementace již existujících řešení pro vyhledávání na základě obsahu.

[15] http://en.wikipedia.org/wiki/Taxicab_geometry

4. Popis implementace

Systém vyhledávání na základě obsahu obrázku, který je popsán v této diplomové práci, se skládá ze tří hlavních procesů: vytvoření databáze (získání obrázků), komprese souborů (vytvoření kódových knih) a vyhledávání podobnosti mezi obrázky. Implementovaný systém pokrývá všechny kroky procesu vyhledávání obrázků. Schéma interakce mezi aplikacemi je zřejmé z *obrázku č 10.* Byly vytvořeny tyto samostatné aplikace:

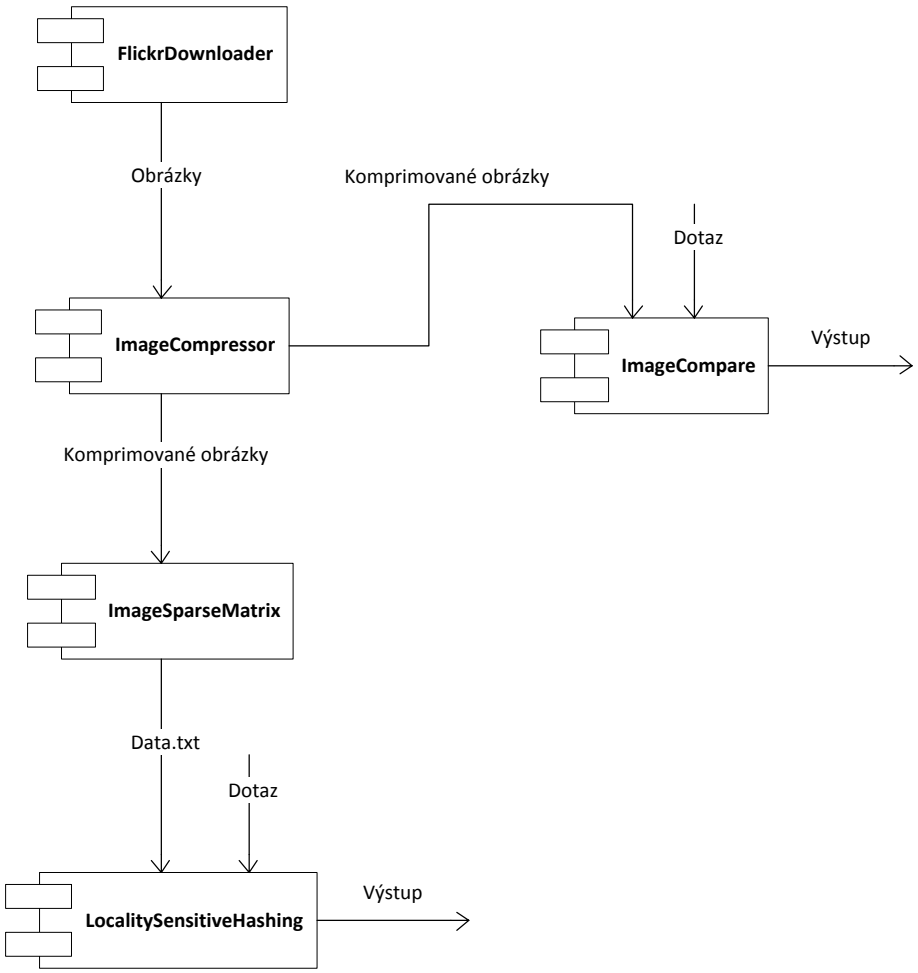
- Konzolová aplikace pro získávání dat (obrázků) ze serveru Flickr do lokálního úložiště (podkapitola 4.1). Obrázky se stahují dle vstupního souboru s klíčovými slovy.

- Konzolová aplikace pro kompresi pomocí vektorové kvantizace a indexaci vstupních obrázků (podkapitola 4.2). Vektorová kvantizace využívá již zmíněné algoritmy K-means a LBG. Výstupem indexace jsou dva soubory: soubor s kódovou knihou a soubor s odkazy do kódové knihy.

- Webová aplikace pro vyhledávání podobných obrázků (podkapitola 4.4). Aplikace získává vlastnosti z kódových knih, které následně porovnává pomocí Hausdorffovy vzdálenosti. Po vypočtení vzdáleností mezi všemi obrázky mohou být obrázky seřazeny podle relevance pro zadaný dotaz. Výsledky jsou vizualizovány prostřednictvím webové stránky.

- Konzolová aplikace pro vytvoření řídké matice (podkapitola 4.5) pro export do projektu *Locality Sensitive Hashing* (dále LSH). Aplikace pouze připraví data, samotné vyhledávání podobných obrázků je realizováno v aplikaci LSH. Naimplementování aplikace LSH nebylo cílem této diplomové práce.

- Konzolová aplikace pro obnovení obrázků z kódové knihy (podkapitola 4.6). Aplikace umožňuje jako výstup zobrazit obrázky po provedení kvantizace a také umožňuje zobrazit pouze kódovou knihu.



Obrázek 10: Schéma jednotlivých aplikací a interakce

4.1. **Aplikace pro sběr dat**

Aplikace slouží ke stahování fotek uložených na serveru Flickr.com. Aplikace je využita pro vytvoření jedné ze dvou částí databáze obrázků pro experimenty. Flickr poskytuje velmi slušně zpracované API[16], které umožňuje získávat informace o fotografiích, uživatelích, skupinách a mnoho dalších. Pro přístup k metodám Flickr API je vyžadována autentizace pomocí standardu OAuth[17]. K používání API je nutný uživatelský účet a dále je nutné aplikaci zaregistrovat. Po zaregistrování se zobrazí klíč a tajná část klíče potřebná k autentizaci. Pro

[16] http://www.flickr.com/services/api/
[17] http://oauth.net/core/1.0a/

usnadnění procesu autentizace jsem se rozhodl využít knihovnu FlickrNet API Library[18], napsanou v jazyce C#.

Instance třídy Flickr sama provede autentizaci po zadání klíče a tajného klíče.

```
Flickr flickr = new Flickr("API_KEY", "SHARED_SECRET");
```

Pro nastavení parametrů vyhledávání na serveru Flickr je třeba vytvořit objekt typu *PhotoSearchOptions*.

```
PhotoSearchOptions searchOptions = new PhotoSearchOptions();
```

Z mnoha parametrů objektu používám především:

- searchOptions.SortOrder – seřazení podle relevance či data odeslání na server;
- searchOptions.PerPage – kolekce výsledků je „stránkována“, určení množství fotografii v jedné stránce;
- searchOptions.Extras – množina informací o fotografii, které chceme získat;
- searchOptions.Tags – nastavení klíčových slov;
- searchOptions.MediaType – typ objektu, fotografie/video;
- searchOptions.UserId – možnost získání fotografií od konkrétního uživatele;
- searchOptions.Latitude – nastavení geolokace, zeměpisná šířka;
- searchOptions.Longitude - nastavení geolokace, zeměpisná délka;
- searchOptions.MinUploadedDate – minimální datum odeslání na server;
- searchOptions.MaxUploadedDate – maximální datum odeslání na server.

Po nastavení parametrů stačí zavolat metodu *PhotosSearch* a výsledek uložit do kolekce *PhotoCollection*.

```
PhotoCollection photoCollection = flickr.PhotosSearch(searchOptions);
```

Aplikace zjišťuje URL adresu jednotlivých obrázků v požadované kvalitě. Vlastnost *MediumUrl* obsahuje URL adresu k obrázku, jehož velikost je dostatečná pro mou testovací databázi (rozměr je 500px na delší straně obrázku). Pomocí třídy *System.Net.WebClient* (jmenný prostor System.Net, ve frameworku .NET) aplikace stáhne a uloží obrázky do lokálního souboru. Podle získaných parametrů ze souboru *kw.txt* aplikace stahuje obrázky relevantní na

[18] http://flickrnet.codeplex.com/

zadaná klíčová slova. V cyklu algoritmus stahování pracuje, dokud uživatel aplikaci neukončí stisknutím tlačítka *Esc*. Obrázky jsou seřazeny podle relevance, pro přístup k stále novým obrázkům se v cyklu zvyšuje parametr stránkování (*searchOptions.Page*). Soubor *kw.txt* obsahuje klíčová slova oddělená novým řádkem.

4.2. Aplikace pro vektorovou kvantizaci a vytvoření kódových knih (indexace obrázků)

Před používáním systému s uživatelskými dotazy do databáze je nutné celou databázi obrázků opatřit indexy. Pomocí indexů budou vyhodnocovány shody podobnosti mezi obrázky. Nejdůležitější parametr je počet průchodů algoritmu LBG, který přímo ovlivňuje počet kódových slov v kódové knize. Hodnota musí být mocninou čísla dvě. Čím je vyšší hodnota parametru, tím lepší bude obrazová kvalita obrázku po kompresi vektorovou kvantizací, ale také se prodlouží doba nutná k indexaci. Parametr rozměr vektorů je nastaven na hodnotu 48, již jsem zmínil, že jeden vektor obsahuje barevnou informaci 4 x 4 obrazových bodů, každý obrazový bod obsahuje 3 barvy RGB prostoru. Protože trénovací vektory jsou zvoleny z vstupních dat ke kompresi, velikost trénovací kódové knihy závisí na velikosti vstupního obrázku. Indexaci jsem prováděl pro dvě různé velikosti vstupního obrázku (originální 512 x 384px a zmenšené 384 x 288px) a s různým nastavením vektorové kvantizace – počtem kódových slov.

Načítání obrázku a jeho jednotlivých barev RGB prostoru do proměnné *list* se provádí v cyklu pomocí metody *GetPixel* , proměnná *sizeOfImage* definuje počet oblastí 4x4pixely v obrázku.

```
List<int> list = new List<int>();
Point sizeOfImage = new Point();
Bitmap bmp = new Bitmap(filename);
...
for (int x = 0; x < sizeOfImage.X; x++) {
  for (int y = 0; y < sizeOfImage.Y; y++) {
    for (int j = 0; j < 4; j++) {
      for (int k = 0; k < 4; k++) {
            list.Add(bmp.GetPixel((x * 4) + k, (y * 4) + j).R);
            list.Add(bmp.GetPixel((x * 4) + k, (y * 4) + j).G);
            list.Add(bmp.GetPixel((x * 4) + k, (y * 4) + j).B);
      }
    }
  }}
```

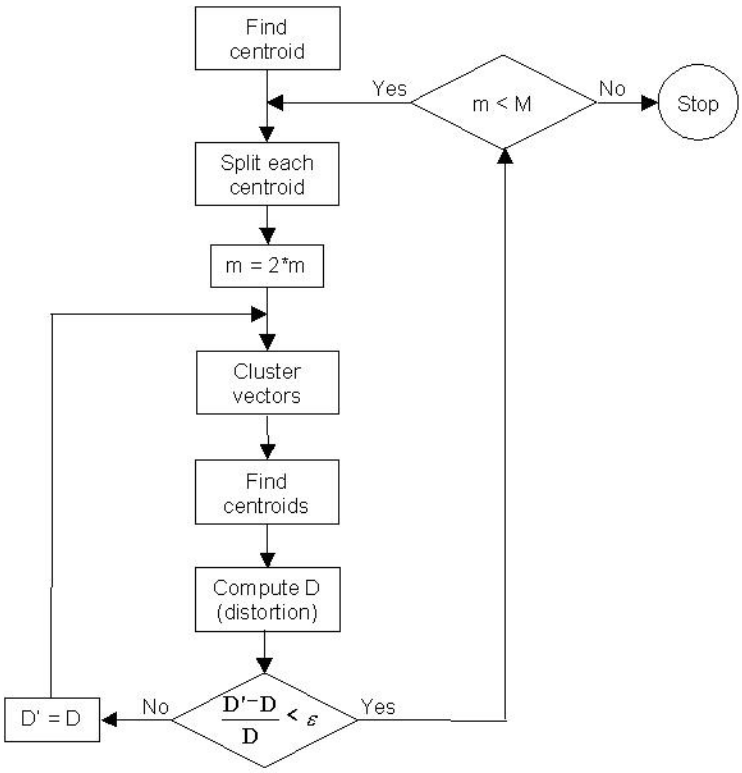
4.3. Implementace LBG, K-means

První krok aplikace je načtení obrázku pro vektorovou kvantizaci a použití obrázků na vstupu k vytvoření trénovacích vektorů. Počet trénovacích vektorů by měl být dostatečně velký, doporučeno je přes 1000 hodnot [15]. Pokud má vstupní obrázek rozměr 384 x 288px počet vektorů, ze kterých se vytvoří trénovací vektory je 6912 hodnot.

V první iteraci se vytvoří *centroid* všech trénovacích vektorů. Tento *centroid* má všechny prvky rovny nule. Poté se v cyklu opakuje proces algoritmu LBG. V každé iteraci se každý vektor v kódové knize rozštěpí na dva nové vektory. Štěpení probíhá tak, že se přičte, resp. odečte hodnota parametru pro odtažení vektoru. Po dokončení štěpení se provede optimalizace kódové knihy, algoritmem k-means, který upraví *centroidy*. Optimalizace se provádí opět v cyklu, dokud se nesplní podmínka, kdy relativní zkreslení je menší, než nastavený parametr *epsilon (obrázek 11)*. Zkoušeli jsme různé hodnoty parametru *epsilon*, avšak nejlepší časové náročnosti dosahoval algoritmus při hodnotě *epsilon* rovné 0,001.

```
private void K_Means() {
  ...
  double result = ((DVQ(0) - DVQ(1)) / DVQ(1));
  if (result <= m_Epsilon) {
      return;
  }
}
```

Po dokončení se výsledná kódová kniha uloží do souboru **.csv*. První dvě hodnoty v souboru jsou rozměry obrázku nutné k následné obnově obrázku z kódové knihy. Poté se už zapisují jednotlivé kódové vektory. Aplikace vytvoří druhý soubor nutný pro obnovu obrázku, tím je soubor s odpovídajícími indexy (odkazy) do kódové knihy. Vektory vstupního obrázku jsou po vektorové kvantizaci převedeny na čísla odkazů do kódové knihy.



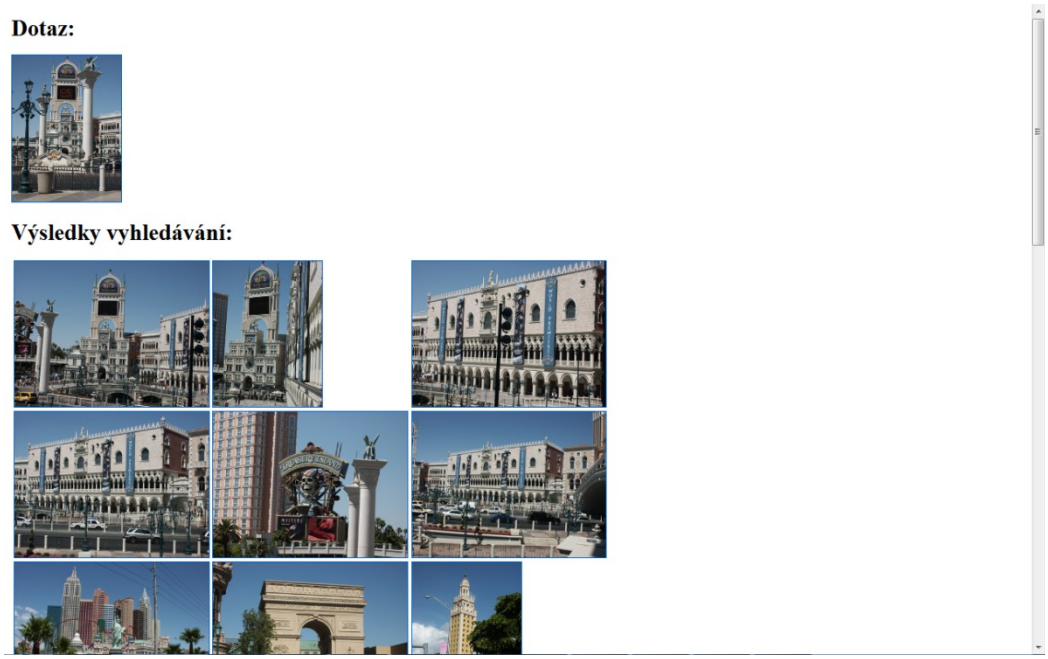
Obrázek 11: Diagram algoritmu vektorové kvantizace [16]

4.4. Vyhledávání obrázků

Aplikace slouží pro vizualizaci výsledků vyhledávání podobných obrázků pro zadaný dotaz. Vstupem pro aplikaci je dotaz reprezentován obrázkem, výstupem je jednoduchá webová stránka, která obsahuje celkem 30 nejrelevantnějších výsledků pro zadaný dotaz.

Aplikace vyhledávání podobnosti využívá modifikovaného Hausdorffova algoritmu a euklidovské vzdálenosti. Program obsahuje i oba další postupy vyhledávání, jak je popsáno v sekci *3.7*. Aplikace počítá podobnost mezi dotazem a obrázky z databáze. Podobnost, reprezentována vzdáleností, slouží jako míra relevance, podle které se řadí výsledky.

Webová stránka obsahuje pouze dvě komponenty, načítání souboru pomocí komponenty *FileUpload* a tlačítko (komponenta *Button*) pro spuštění vyhledávání. Po stisknutí tlačítka *„Vyhledej*“ se provede kontrola, jestli je komponentou *FileUpload* vybrán soubor. Pokud ano, vytvoří se instance třídy *Hausdorff* a provedou se dvě její metody *CompareBy* a *PrintResult*.



Obrázek 12: Ukázka výstupního souboru po vyhledání na dotaz

Struktura dat

- Default.aspx – webová stránka s UI pro vybrání souboru označeného jako dotaz;
- Hausdorff.cs – třída s algoritmy pro porovnávání;
- App_Data\db\ - adresář, obsahující soubory s kódovými knihami **.csv;*
- App_Data\img\ - adresář, obsahující obrázky **.tif, *.jpg*.

Upravil jsem také soubor *web.config* změnou v parametru *executionTimeout*. Nyní webový prohlížeč nebude uvádět chybu v případě, že by porovnávání obrázku překočilo povolenou odezvu 110 vteřin.

```
<httpRuntime executionTimeout="2000"/>
```

4.5. Implementace výstupu řídké matice

Jako možná alternativa k poměrně pomalému vyhledávání s využitím Hausdorffova algoritmu bylo vyzkoušet aplikaci *Locality Sensitive Hashing* (dále LSH). Bylo nutné vytvořit textový soubor s vstupními daty a vyzkoušet funkčnost aplikace LSH. Vstupní soubor má následující schéma:

```
id;column;value;column;value...
id;column;value;column;value...
...
```

Parametr	Význam
id	id, či název obrázku
column	identifikátor kódového slova, číslováno od nuly, automatické zvýšení o 1, pokud se jedná o nové kódové slovo
value	počet, kolikrát obrázek obsahuje dané kódové slovo

Tabulka 1: Význam parametrů ve vstupním souboru pro LSH

Při takovém schématu může nastat velké množství možných kódových slov a pouze ta nejpodobnější z nich se budou opakovat (sloupec *column*). Protože dva naprosto identické vektory o rozměru 48 prvků by se objevily jen velmi těžko, naimplementoval jsem seskupování kódových slov. Kódová slova tedy nemusí být pro využití již zaindexovaného kódového slova identická, stačí, pokud je vzdálenost mezi nimi menší než parametr pro toleranci seskupování. Tento parametr byl nastaven na hodnotu 100. Pokud více vektorů splňuje zmíněnou podmínku, vybere se vektor s nejmenší vzdáleností.

Pro každý obrázek algoritmus prochází jeho kódovou knihu a pro všechna kódová slova zjišťuje, jestli už bylo kódové slovo (či podobné) použito. Pokud bylo použito, vloží se do parametru *column* číslo již existující kódové knihy, pokud je vzdálenost mezi vektory větší, do proměnné se vloží nové kódové slovo a vytvoří se nový index. K patřičné hodnotě column se přidá parametr *value*, který značí, kolikrát bylo kódové slovo v obrázku použito. Použitá kódová slova a jejich index se uchovávají pouze jako lokální proměnné a po ukončení aplikace se dále neuchovávají. Pro výpočet vzdálenosti pro seskupování jsem využíval opět Hausdorffovu vzdálenost, výhoda však je, že se takto data připraví jednou pro celou databázi obrázků a pro dotaz se již pomalá Hausdorffova vzdálenost nevyužívá.

Kód výpočtu hodnoty parametru *value* (kolikrát bylo v obrázku zastoupeno dané kódové slovo):

```
StreamReader sr = new StreamReader(filename);
for (int i = 0; i < sizeOfCodebook; i++) {
      codeWord[i] = 0;
}
while (!sr.EndOfStream) {
      no = int.Parse(sr.ReadLine().ToString());
      codeWord[no]++;
}
```

4.6. Aplikace pro obnovení obrázků z kódové knihy

Pro představu, jak vypadá obrázek po vektorové kvantizaci, jsem vytvořil samostatnou aplikaci, která pro vstupní kódovou knihu a soubor s indexy vytvoří, zrestauruje obrázek. První dva řádky souboru s kódovou knihou obsahují výšku a šířku původního obrázku. Tyto hodnoty potřebuji k zapsání jednotlivých vektorů na správně umístění. Aplikace pracuje tak, že v souboru s odkazy do kódové knihy nahradí index vektoru v kódové knize přímo vektorem z kódové knihy. Obnova obrázku trvá cca 1 vteřinu. Výsledky této aplikace – komprimované obrázky s různým nastavením komprimace jsem použil v textu diplomové práce. Na obrázku *č 13* je zobrazen komprimovaný obrázek včetně kódové knihy (32 kódových slov - vzorů), pomocí které je složen.

Obrázek 13: Komprimovaný obrázek a jeho kódová kniha

5. Experimenty

Cílem této kapitoly je ověření funkčnosti jednotlivých aplikací a také získání výsledků při různých nastaveních aplikací s cílem najít nejefektivnější konfigurace. První dvě podkapitoly popisují samotná vstupní data a jejich získání pro provádění dalších experimentů. V podkapitole 5. 3 je sledována časová náročnost a účinnost komprese, resp. poměr velikosti souborů před a po vektorové kvantizaci. V podkapitole 5. 4 jsou za hlavní parametry relevance vyhledávání zvoleny parametry přesnosti, úplnosti a f-míry, které odráží dosaženou efektivitu systému vyhledávání a porovnávání obrázků.

5.1. Popis vstupních dat

Pro experimenty s vyhledáváním a porovnáváním obrázků bylo potřeba zajistit potřebnou databázi obrázků. Ideální databáze obrázků pro srovnávací test by měla být roztřízena do složek se sobě korespondujícími (podobnými) obrázky nebo by měl existovat textový soubor tzv. „*ground truth*“, který popisuje, které obrázky k sobě korespondují po provedení výpočtu podobnosti. Pro testování je také důležité, abychom testy prováděli na stále stejné testovací sadě obrázků. Byly vytvořeny dvě oddělené testovací sady obrázků:

- Pro vizualizaci výsledků na náhodných obrázcích jsem použil obrázky stažené ze serveru Flickr. Nevýhoda je, že stažené obrázky jsou ve formátu **.jpg*, tedy již na nich byla provedena komprese.
- Pro testování výsledků efektivity vyhledávání jsem využil databázi nekomprimovaných obrázků UCID dostupnou na webu univerzity v Loughborough.

Tato data dává k dispozici Dr. Gerald Schaefer, který je zároveň autorem článku Content-based Retrieval of Compressed Images [12]. Data v UCID (Uncompressed Colour Image Database, UCID - výslovnost "use-it") jsou v rozlišení 640x480 pixelů a formátu TIFF. V aktuální verzi databáze je přes 1300 obrázků. Databáze také obsahuje textový soubor *ground truth.*

V dostupných zdrojích na internetu jsou obvykle obrázky velkých rozměrů, avšak pro rychlejší indexaci obrázků bylo provedeno zmenšení získaných obrázků při zachování poměru stran. Vstupní obrázky z kolekce UCID jsou pro účely testování zmenšeny na velikost 384 x

288 pixelů. Provádět kompresi lze i na obrázcích s vysokým rozlišením, avšak je třeba také přenastavit proměnné pro výpočet vektorové kvantizace.

5.2. Stahování obrázků

Aplikace pro stahování obrázku ze serveru Flickr zpracovávala soubor s klíčovými slovy, který obsahoval pět výrazů: *„prague, beach, nature, summer, architecture“*. Aplikace stáhla 703 obrázků za 10 minut. Celková velikost souborů v databázi byla 96,4 MB. Za jeden den je tedy možné stáhnout přibližně 100 tisíc obrázků o velikosti 13,5GB. Takto lze aplikaci nechat spuštěnou na serveru a neustále rozšiřovat lokální databázi obrázků.

5.3. Komprese obrázků

Vyhledávání a porovnávání obrázků je v diplomové práci založeno na komprimovaných datech, proto bylo potřeba vstupní data komprimovat a vytvořit tak kódové knihy. Vytvářel jsem kódové knihy s různým nastavením komprese. Hlavními testovanými parametry byla časová náročnost vektorové kvantizace a velikost souborů po kompresi.

Vektorovou kvantizaci jsem pro porovnání prováděl pro 4 až 7 iterací algoritmu LBG, vlivem iterací se počet kódových slov N v kódové knize zvyšoval od 16 po 128. Po provedení vektorové kvantizace jsou výsledky na kvalitě testovaného obrázku naprosto zřejmé především na počtu segmentů ze kterých se skládá obloha a v zobrazení obličeje sfingy viz. *obr. 13*.

Obrázek 13: Obrázek před a po vektorové kvantizaci, velikost kódové knihy 64

V *tabulkách č.2* a *č.3* byly sledovány parametry velikosti vytvořených souborů a časovou náročnost provedení vektorové kvantizace. Velikost souboru s indexy obsahuje stále stejný počet hodnot. Avšak v důsledku většího počtu kódových slov v kódové knize jsou častěji čísla dvouciferná či trojciferná, proto velikost souboru se zvyšováním počtu kódových slov roste, avšak pouze minimálně. Velikost souboru s kódovou knihou roste přímou úměrou se zvyšováním počtu kódových slov.

Při použití vektorové kvantizace s 128 kódovými slovy se zmenší obrázek z 577kB do dvou souborů s celkovou velikostí 75kB, tzn. komprimovaný obrázek zabírá pouze 13 % původní velikosti.

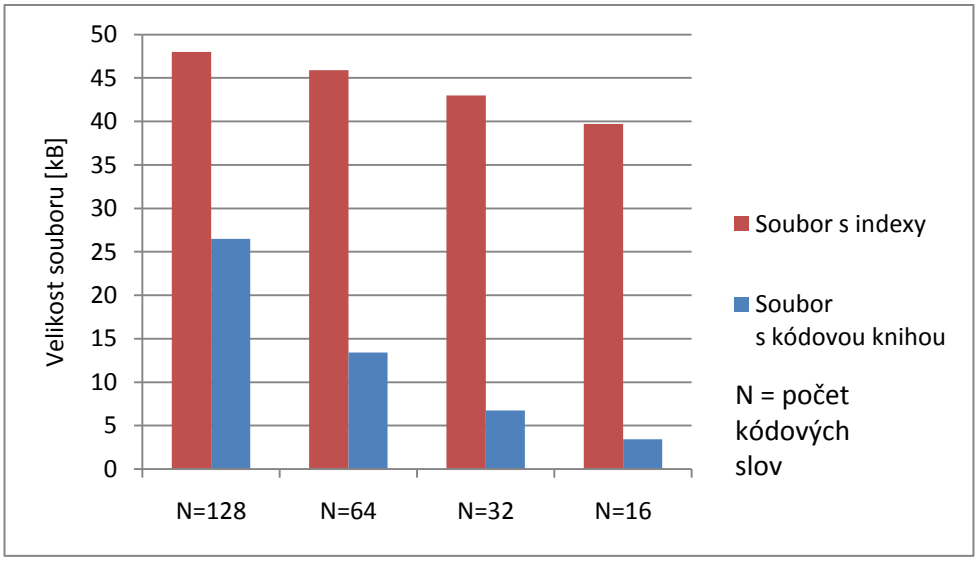
Počet kódových slov	Velikost souboru s kódovou knihou [kB]	Velikost souboru s indexy [kB]	Velikost celkem [kB]	Čas zpracování [s]
128	26,7	26,9	53,6	632
64	13,4	25,7	39,1	267
32	6,76	24,1	30,86	125
16	3,39	22,1	25,49	57
Bez komprese	-	-	346	-

Tabulka 2: Pro obrázek velikosti 346kB, rozměr 384x288

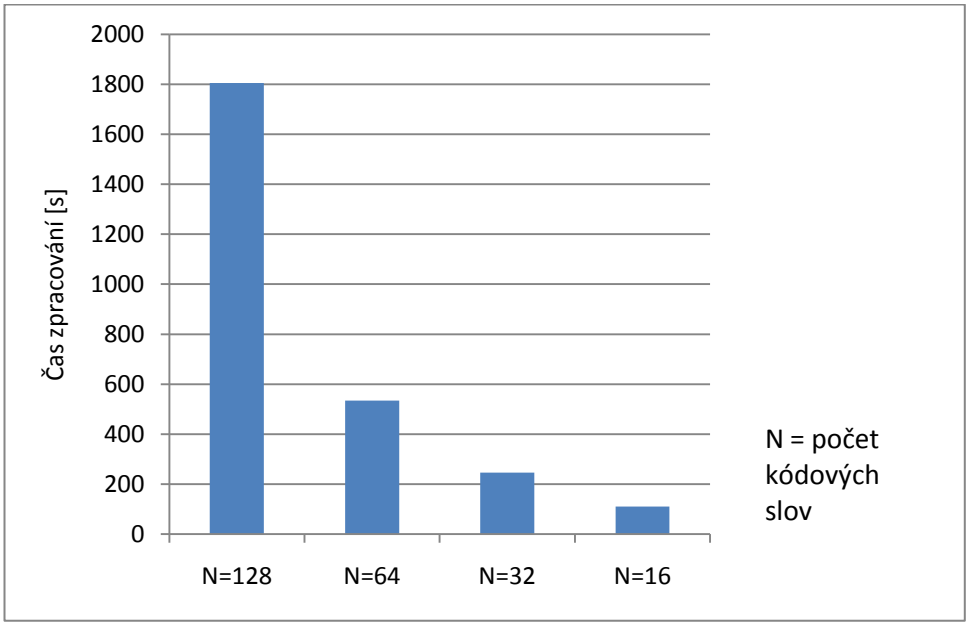
Počet kódových slov	Velikost souboru s kódovou knihou [kB]	Velikost souboru s indexy [kB]	Velikost celkem [kB]	Čas zpracování [s]
128	26,5	48	74,5	1805
64	45,9	45,9	59,3	534
32	6,75	43	49,75	246
16	3,42	39,7	43,12	111
Bez komprese			577	

Tabulka 3: Pro obrázek velikosti 577kB, rozměr 512x384

Z tabulky č. 3 je patrné, že zvýšením počtu iterací, se čas zvýšil až exponenciálně, pro velikost kódové knihy s 256ti kódovými slovy trval několik desítek minut, proto jsem tuto velikost neuváděl v tabulce. Řešením by bylo v implementaci upravit paramemetr pro relativní zkreslení epsilon a parametr pro odtažení vektorů.



Obrázek 14: Graf velikosti souborů po kompresi



Obrázek 15: Čas indexace obrázku

5.4. **Relevance vyhledávání**

Měření relevance vyhledávání je založeno na porovnávání získaných výsledků s počtem správných výsledků (*ground truth*) a velikosti databáze. Relevance znamená významnost či důležitost, obvykle ve vztahu k nějakému cíli. V mém využití znamená relevance vizuální podobnost dotazu a výsledku dotazu. Ideální vyhledávácí systém vyhledá všechny relevantní dokumenty z databáze a na výstupu nevrátí žádný nerelevantní výsledek. Pro vyjádření relevance používám dvě míry: přesnost a úplnost.

Přesnost (ang. Precision) udává, jestli jsou nalezeny pouze relevantní informace, tzn. poměr mezi nalezenými relevantními objekty a všemi nalezenými objekty. Přesnost měří míru šumu (nerelevantních informací) ve vyhledávání.

$$přesnost = \frac{|\{relevantní\ dokumenty\} \cap \{nalezené\ dokumenty\}|}{|\{nalezené\ dokumenty\}|}$$

Úplnost, výtěžnost (ang. Recall) udává, jestli sou nalezeny všechny relevantní informace, tzn. poměr mezi nalezenými relevantními objekty a všemi relevantními objekty. Úplnost jiným slovy měří, jak důkladné je vyhledávání.

$$úplnost = \frac{|\{relevantní\ dokumenty\} \cap \{nalezené\ dokumenty\}|}{|\{relevantní\ dokumenty\}|}$$

Vztah mezi přesností a úplností je nepřímá úměrnost. Čím vyšší je úplnost, tím nižší je přesnost a naopak, tedy pokud jsou všechny výsledky relevantní, lze předpokládat, že některé další relevantní výsledky odpovídající na dotaz chybí. Proto se snažíme o vyrovnané hodnoty obou charakteristik. Lépe je pro vyhodnocování použít jednu hodnotu, tzv. harmonický průměr přesnosti a úplností, tou je tzv. f-míra (ang. f-measure, f-score) .

$$F = 2\frac{úplnost * přesnost}{úplnost + přesnost}$$

Porovnávání jsem prováděl pomocí přesnosti, účinnosti a F-míry. Měření jsem prováděl na *UCID* datech, protože je u nich definován *ground truth* soubor. Databáze stažených fotek z Flickru nelze použít, protože jediný způsob jak definovat, které obrázky jsou si podobné a které ne, lze, pouze pokud bychom dopředu určili korespondující obrázky, což by bylo časově velmi náročné a nepřesné.

Vyzkoušel jsem několik algoritmů porovnávání kódových knih a měřil výsledky:

- Modifikovaná jednostranná Hausdorffova vzdálenost pro porovnání vektorů, euklidovská vzdálenost pro porovnání jednotlivých prvků vektoru, definována jako

$$d_H(A,B) = h(A,B)$$

- Modifikovaná oboustranná Hausdorffova vzdálenost pro porovnání vektorů, euklidovská vzdálenost pro porovnání jednotlivých prvků vektoru

$$d_H(A,B) = \max\big(h(A,B), h(B,A)\big)$$

- Modifikovaná jednostranná Hausdorffova vzdálenost pro porovnání vektorů, Manhattanská vzdálenost pro porovnání jednotlivých prvků vektoru, definována jako

$$d_H(A,B) = h(A,B)$$

- Využití aplikace Locality Sensitive Hashing.

V *tabulce č. 4* mají přesnost a účinnost vždy stejnou hodnotu, jelikož jsem za počet získaných (vyhledaných) obrázků označil počet obrázků definovaných v *ground truth* jako podobné. Jelikož jsem předem věděl, že např. 5 obrázků v databázi je označeno v *ground truth* jako relevantní, vyhodnocoval jsem pouze prvních 5 obrázku získaných z výstupu aplikace pro porovnávání. Vyhledávání jsem opakoval pro 20 vybraných obrázků, stejné obrázky jsem použil i pro *tabulku č. 5.*

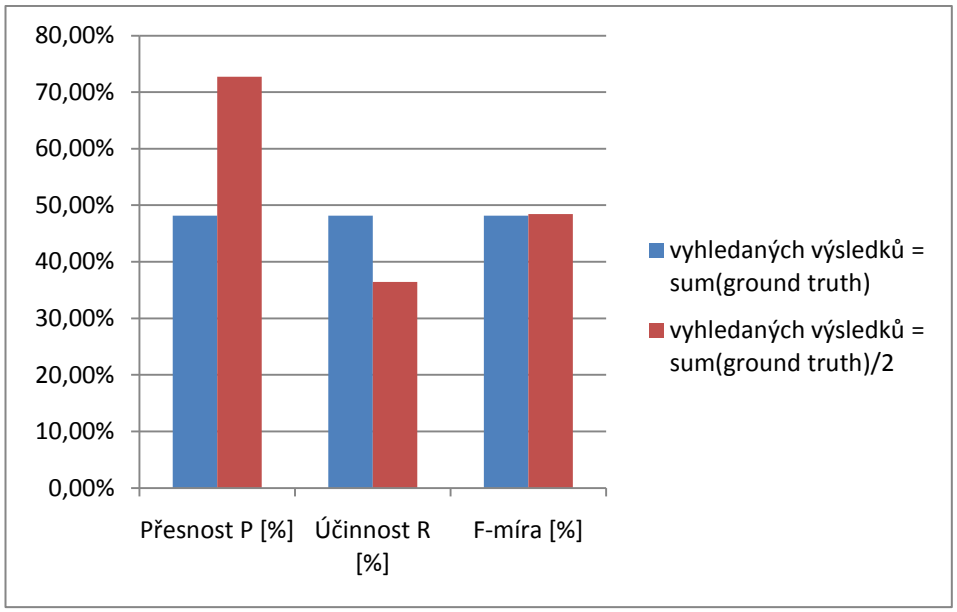
Popis	Přesnost P [%]	Účinnost R [%]	F-míra [%]
d(A,B)=h(A,B); euklidovská vzdálenost	48,13	48,13	48,13
d(A,B)=max (h(A,B),h(B,A)); euklidovská vzdálenost	42,06	42,06	42,06
d(A,B)=h(A,B); manhattanská vzdálenost	42,43	42,43	42,43
Locality Sensitive Hashing	20,88	20,88	20,88

Tabulka 4: Přesnost, účinnost, f-míra

Pro následující měření efektivity vyhledávání (*tab. 5*) jsem test prováděl pouze s polovinou *ground truth*, tzn., je-li 6 obrázků v *ground truth* označeno za podobné, vyhodnocoval jsem první tři výsledky zobrazené aplikací. Zvýšila se přesnost vyhledávání, za cenu snížení účinnosti (některé relevantní obrázky pro dotaz aplikace nezobrazila), *f-míra* zůstala na přibližně stejných hodnotách jako v předchozím měření.

Popis	Přesnost P [%]	Účinnost R [%]	F-míra [%]
d(A,B)=h(A,B); euklidovská vzdálenost	72,73	36,44	48,46
d(A,B)=max (h(A,B),h(B,A)); euklidovská vzdálenost	60,25	29,99	39,96
d(A,B)=h(A,B); manhattanská vzdálenost	61,25	30,51	40,71
Locality Sensitive Hashing	31,36	16,10	21,26

Tabulka 5: Přesnost, účinnost, f-míra



Obrázek 16: Vliv počtu vyhledaných výsledků na přesnost a účinnost

Výsledky závisí na kvalitě a obsáhlosti *ground truth* souboru, reálné výsledky efektivity aplikace mohou dosahovat mírně lepších hodnot. Dle mého názoru aplikace v mnoha případech vyhledala podobné obrázky, které bych subjektivně označil za relevantní, avšak v *ground truth* souboru, který jsem měl k dispozici, tyto obrázky nebyly uvedeny, proto jsem je ve svém měření nepočítal jako relevantní.

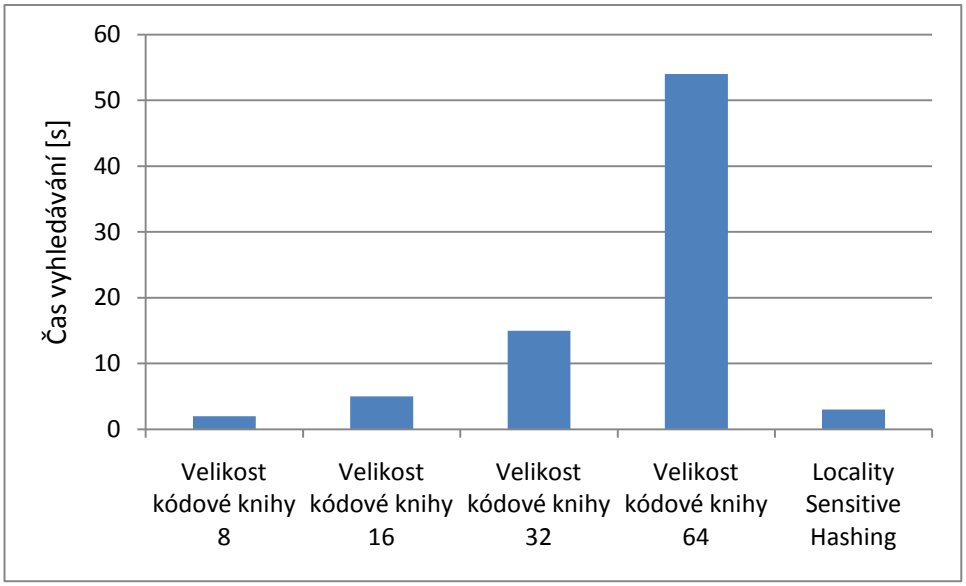
Jako nejlepší metoda v mém testování vyšla jednosměrná Hausdorffova vzdálenost, provedená pouze jako $d_H(A,B) = h(A,B)$, nikoliv pomocí obousměrné Hausdorffovy vzdálenosti $d_H(A,B) = \max\big(h(A,B), h(B,A)\big)$, jak bylo popsáno v článku G. Schaefera.

Pro vysvětlení uvedu příklad: máme dva podobné obrázky – jeden jako dotaz (dále obrázek A), druhý jako obrázek k porovnání vybraný z databáze (dále obrázek B). Na obrázcích je postava na pláži, moře a obloha. Snímek z databáze je vizuálně velmi podobný dotazu, avšak s rozdílem, že byl pořízen více proti slunci a část oblohy je tzv. přepálená (barva přibližující se bílé barvě, v RGB zápise (255,255,255). Na snímku položeném jako dotaz je vše správně, obloha je čistě modrá (RGB zápis (60,130,180))

Pokud hledám vektor s částí modré oblohy z obrázku A naleznu jej v části modré oblohy B, tedy vzdálenost vektorů je minimální. Ale naopak pokud hledám vektor bílé oblohy z obrázku B, nenaleznu jej v obrázku A, resp. naleznu, avšak vzdálenost mezi vektory bude velmi velká, což zásadně změní průměr vzdáleností h(B,A), obzvlášť když například obloha zabírá třetinu obrázku. Výsledek Hausdorffovy vzdálenosti je maximum z vzdáleností h(A,B) a h(B,A), tedy právě h(B,A), i přestože vzdálenost h(A,B) je malá, protože ke všem vektorům z obrázku A byl nalezen vektor z obrázku B s minimální vzdáleností, tedy velkou shodou mezi obrázky.

	Velikost kódové knihy 8	Velikost kódové knihy 16	Velikost kódové knihy 32	Velikost kódové knihy 64	Locality Sensitive Hashing
Čas vyhledávání v 1338 souborech (UCID) [s]	2	5	15	54	3

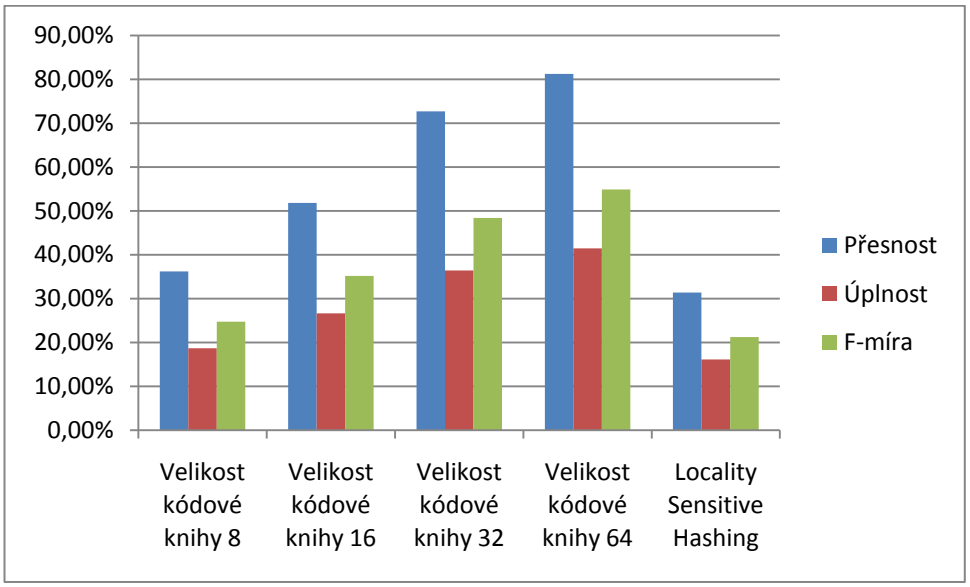
Tabulka 6: Časová náročnost vyhledávání



Obrázek 17: Časová náročnost vyhledávání

	Velikost kódové knihy 8	Velikost kódové knihy 16	Velikost kódové knihy 32	Velikost kódové knihy 64	Locality Sensitive Hashing
Přesnost [%]	36,22	51,77	72,7	81,21	31,36
Úplnost [%]	18,66	26,65	36,44	41,43	16,10
F-míra [%]	24,71	35,17	48,4	54,85	21,26

Tabulka 7: Parametry efektivity vektorové kvantizace



Obrázek 18: Měření efektivity v závislosti na míře komprese

5.5. Zhodnocení experimentů

Výsledky relevance vyhledávání pomocí aplikace *Locality Sensitive Hashing* byly poměrně slabé (pouze 31% přesnost), naopak rychlost vyhledávání byla velmi dobrá. Relevance výsledků by pravděpodobně šla zlepšit dlouhodobějším testováním pro zjištění nejlepšího formátu vstupních dat a nastavení parametrů aplikace *LSH*.

Relevance vyhledávání velmi záleží na nastavení vektorové kvantizace. Čím větší je míra komprimace, tím více jsou nepřesné informace, které z komprimovaných dat lze získat. Nejlepší hodnoty dosahuje algoritmus při velikosti kódové knihy 64 kódových slov, přesnost přesáhla hodnotu 80 %. Je nutné si uvědomit, že zmíněná hodnota relevance vyhledávání je dosažena při komprimaci na cca 10% velikost originálních obrázků. Lze očekávat, že při menší komprimaci by efektivita vzrostla. Větší kódovou knihu jsem však netestoval, kvůli časové náročnosti vytváření těchto kódových knih.

Pokud bychom se snažili urychlit proces vyhledávání, je třeba rozdělit proces do dvou částí – indexaci a samotné vyhledávání. Nevýhoda algoritmu K-means, použitého při indexaci je, že jej nelze provádět v reálném čase, což ale nemusí být problém, jelikož server bude mít aplikaci spuštěnou na pozadí a bude průběžně indexovat nově vložené obrázky bez vědomí uživatele. Po zadání uživatelského dotazu se provádí proces porovnávání kódových knih. Pro zrychlení vyhledávání by pomohlo nahradit sekvenční přístup k datům za datovou strukturu R-strom, či některou z jeho variant. Jako další možnost zrychlení vyhledávání se nabízí reprezentovat obrázek pouze několika vektory s nejčastějším výskytem. Zkrátila by se tak časová odezva od položení dotazu, ale zvýšila by se nepřesnost výsledků.

5.6. Testovací podmínky

Všechny testy jsem prováděl na notebooku s 2 GHz Intel Core 2Duo procesorem, 2GB RAM a 5400 otáčkovým pevným diskem, s operačním systémem Windows 7. Rychlost internetového připojení byla přibližně 14 Mbit/s.

6. Závěr

Cílem mé práce bylo nastudovat odborný článek G. Schaefera a implementovat jednu z popsaných metod vyhledávání obrázků na základě obsahu z komprimovaných dat. Pro vizualizaci výsledků vyhledávání obrázků jsem vytvořil databázi pomocí aplikace pro stahování obrázků ze serveru Flickr. Testování efektivity vyhledávání jsem prováděl na datech *UCID*, kvůli již existujícího *ground truth* souboru. V mé implementaci jsem dosáhl více než 80-ti procentní přesnosti ve výsledcích vyhledávání. Tento výsledek však závisí na mnoha faktorech, jako je míra komprese vstupních obrázků, nastavení účinnosti (*recall*) při měření a v neposlední řadě na kvalitě souboru s *ground truth*. Při použití obrázků z databáze Flickr se provádí komprimace vektorovou kvantizací na již komprimovaných souborech (JPEG komprese), proto se efektivita může mírně snížit.

V průběhu práce jsem studoval možné využití kvaternionů pro uložení tří hodnot barvy (barevný prostor RGB) jednoho obrazového bodu. Kvaternion je nekomutativní rozšíření komplexních čísel. Mezi kvaterniony a čtyřrozměrnými vektory je principiální rozdíl - operace dělení je mezi dvěma kvaterniony definována, zatímco mezi dvěma vektory tato operace vůbec neexistuje. Použití kvaternionu mohlo přinést úsporu velikosti vektorů a zefektivnění komprese, díky využití operací násobení a dělení na kvaternionech. Bohužel tyto operace v průběhu vektorové kvantizace nevyužívám, tedy využití kvaternionů by oproti čtyřrozměrným vektorům nepřineslo patřičné výhody.

Na webu dnes stále převládají systémy pro vyhledávání využívající textovou anotaci. K zefektivnění systému by přispělo přidat prvky ze zmíněného přístupu vyhledávání na základě textové anotace, např. popis pomocí několika klíčových slov nebo geolokaci, kterou server Flickr poskytuje k dispozici, pokud ji obrázek obsahuje. Geolokační služby jsou na vzestupy a téměř každý nový *smartphone* obsahuje GPS a umožňuje k fotografii přidat zeměpisné údaje o místě zachycení.

Z metod pro vyhledávání na základě obsahu by bylo možné rozšířit systém o segmentaci obrázku pomocí prahování, hledání hran či hledání oblastí. Avšak vektorová kvantizace naruší prostorové uspořádání původního obrázku, proto se využívá pouze pro porovnávání na základě barevnosti a textury. Segmentaci obrazu využívá druhá metoda popsaná

v části textu G.Schafera a tou je vyvinutí vlastního čtyř-kritériového kompresního algoritmu, ze kterého jsou data vizuálně čitelná i v komprimované podobě.

Při vývoji systému pro vyhledávání na základě obsahu obrázků musíme také dbát na pochopení dotazu uživatele. Pokud chceme uživateli prezentovat nejlepší výsledky, potřebujeme sledovat chování uživatele při vyhledávání, jako například, jak lidé popisují obrázky či jiné objekty. Proto je oblast získávání informací (*information retrieval*) velice široký mezivědní obor. Steve Jobs prohlásil „*Lidé často nevědí, co chtějí, dokud jim to neukážete*“, pokud bych tuto větu převedl na systémy pro vyhledávání obrázků, zaměřil bych se nejen na matematicky efektivní a dokonalé algoritmy pro indexaci a porovnávání, ale také na snahu o vyvinutí co nejlepšího uživatelského rozhraní, kde by uživatel mohl snadno „vymodelovat“ dotaz, který nejlépe odpovídá tomu, co opravdu hledá.

Pro případné navázání na mou diplomovou práci bych doporučil vytvořit přehledné grafické rozhraní umožňující zadat dotaz v podobě náčrtku. Grafické rozhraní by také mohlo obsahovat výběr určitých barev a odhad jejich procentuálního zastoupení v obrázku. Z takto definovaného dotazu by se vytvořil obrázek a proběhlo by porovnání podobnosti s obrázky v databázi.

Pravděpodobně nelze udělat dokonalou aplikaci pro vyhledávání vhodnou pro všechna použití, např. vyhledávání podobnosti otisků prstů bude těžko využívat porovnávání podle barevnosti, ale bude založeno na prahování či detekci hran. Řešení je najít pro každé odvětví tu nejefektivnější kombinaci postupů a metod.

Literatura

[1] BENEŠ, Miroslav. – ZITOVÁ Barbara. *Nephele: Databáze restaurátorských zpráv s možností vyhledávání podle textové a obrazové informace*. [online]. 2005 [cit. 2012-04-09]. Dostupné z: <dar.site.cas.cz/download.php?bd=247>

[2] BLAŽEK, Jakub. *Systémy vyhledávání obrazových informací. Část II.: Problematika vyhledávání.* [online]. 2010 [cit. 2012-04-09]. Dostupné z: <http://www.inflow.cz/systemy-vyhledavani-obrazovych-informaci-cast-i-problematika-vyhledavani>. ISSN 1802-9736.

[3] *CBIR: Texture Features*. [online]. [cit. 2012-04-09]. Dostupné z: <http://www.cs.auckland.ac.nz/compsci708s1c/lectures/Glect-html/topic4c708FSC.htm>

[4] ČERNOCKÝ, Jan. *Vektorové kvantování*. [online]. [cit. 2012-04-09]. Dostupné z: <http://www.fit.vutbr.cz/~cernocky/oldspeech/lectures/vq.pdf>

[5] DOBEŠ, Michal. *Zpracování obrazu a algoritmy v C#.* vyd. Praha: BEN, 2008. 144 s. ISBN 978-80-7300-233-6.

[6] EAKINS, John. *Content-based Image Retrieval*. [online]. 1999 [cit. 2012-04-09]. Dostupné z: <http://www.jisc.ac.uk/media/documents/programmes/jtap/jtap-039.pdf>

[7] GRÉGOIRE, Normand - BOUILLOT Mikael. *Hausdorff distance between convex polygons*. [online]. 1998 [cit. 2012-04-09]. Dostupné z: <http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html>

[8] *K-Means Algorithm.* [online]. [cit. 2012-04-09]. Dostupné z: <http://www.kovan.ceng.metu.edu.tr/~maya/kmeans/other.html>

[9] KUČEROVÁ, Helena. *Kvantitativní metody zkoumání dokumentové komunikace*. [online]. 2011 [cit. 2012-04-09]. Dostupné z: <http://info.sks.cz/users/ku/ZIZ/kvant.htm>

[10] SCHAEFER, G. *Content-based retrieval from image databases: Colour, compression, and browsing*. International Conference on Information Retrieval & Knowledge Management CAMP. [online]. 2010 [cit. 2012-04-09]. Dostupné z: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5466891> ISBN 978-1-4244-5650-5

[11] SCHAEFER, G. *Content-based Retrieval of Compressed Images.* [online]. [cit. 2012-04-09]. Dostupné z: <http://ceur-ws.org/Vol-567/invited2.pdf>

[12] SCHAEFER, G. – STICH M. *UCID - An Uncompressed Colour Image Database.* San Jose, USA. 2004. 600 s. ISBN 9780819452108

[13] ŠOCHMAN, Jan. *Cvičení z RPZ - Shlukování k-means* [online]. 2005 [cit. 2012-04-09]. Dostupné z: <http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/kmeans/kmeans.pdf>

[14] ŠPANĚL, M. *Klasifikace a rozpoznávání.* [online]. 2009 [cit. 2012-04-09]. Dostupné z: <http://www.fit.vutbr.cz/study/courses/IKR/public/stare_prednasky_2011/08_clustering/IKR_Clustering.pdf>

[15] *Vector Quantization.* [online]. [cit. 2012-04-09]. Dostupné z: <http://www.data-compression.com/vq.shtml>

[16] *Voice recognition using DSP* [online]. [cit. 2012-04-09]. Dostupné z: http://azhar-paperpresentation.blogspot.com/2010/04/voice-recognition-using-dsp.html

[17] *Wavelety: druhá část.* [online]. 2010 [cit. 2012-04-09]. Dostupné z: <http://zoi.utia.cas.cz/files//prednaska10_w2.ppt>

[18] *Wikipedia: Eukleidovský prostor* [online]. [cit. 2012-04-09]. Dostupné z: <http://cs.wikipedia.org/wiki/Eukleidovský_prostor>

[19] *Wikipedia: Hašovací funkce* [online]. [cit. 2012-04-09]. Dostupné z: <http://cs.wikipedia.org/wiki/Hašovací_funkce>

[20] *Wikipedia: Image compression* [online]. [cit. 2012-04-09]. Dostupné z: <http://en.wikipedia.org/wiki/Image_compression>

Přílohy

Obrázek 19: Originální obrázek

Obrázek 20: Obrázek komprimovaný VQ s 128 kódovými slovy

Obrázek 21: Obrázek komprimovaný VQ s 64 kódovými slovy

Obrázek 22: Obrázek komprimovaný VQ s 32 kódovými slovy